Apple Assembly Line

$1.80

Volume 6 -- Issue 6 March, 1986

ES-CAPE

Whatever happened to the Extended S-C Applesoft Program Editor? That's a question we've heard more than a few times in the last year or two, and we finally have some kind of answer.

We got bogged down in producing Version 2.0 of the program. The new printer control, Park and Join, and Applesoft and DOS command features are great. The 40-column, STB-80, and //e versions came out just fine, but the Videx and Viewmaster versions stumped us. The planned Renumber and Merge features never made it, and we couldn't settle on a mechanism for adding other utility programs.

Anyway, we've got a deal for you! How's this for a package?: ES-CAPE 1.0 Source and Object Code and manual, along with ES-CAPE 2.0 Source and Object Code and a manual supplement on disk. That's all the source and object code for both versions of the program, for a total of only $50.00. Registered owners of ES-CAPE 1.0 can purchase this new package for only $30.00.

New 65816 Book

There's another book coming along on programming the 658xx processors. This one is called "65816/65802 Assembly Language Programming", by Michael Fischer, published by Osborne/McGraw-Hill as an addition to their Assembly Language Programming series, mostly by Lance Leventhal. Fischer's book is scheduled for May delivery, so we have ordered some copies and are beginning to accept orders. Our price will be $18.00 + shipping.

Modifying ProDOS for Non-Standard ROMs...Bob Sander-Cederlof

We have published several times ways to defeat the ROM Checksummer that is executed during a ProDOS boot, so that owners of Franklin clones (or even real Apples with modified monitor ROMs) could use ProDOS-based software. See AALs of March and June, 1984.

Both of these previous articles are out of date now, because they apply to older versions of ProDOS than are current. What follows applies to Version 1.1.1 of ProDOS.

There are two problems with getting ProDOS to boot on a non-standard machine. The first is the ROM Checksummer. This subroutine starts at $267C in Version 1.1.1, and is only called from $25EE. The code is purposely weird, designed to look like it is NOT checking the ROMs. It also has apparently purposeful side effects. Here is a listing of the subroutine:

```
                 1000 *SAVE CHECKSUMMER
                 1010 *--------------------------------
                 1020        .OR $267C    POSITION IN PRODOS SYSTEM FILE
                 1030 *--------------------------------
                 1040 CHECKSUMMER
267C- 18         1050        CLC
267D- AC 74 26   1060        LDY $2674    (GETS A VALUE 0)
2680- B1 0A      1070 .1     LDA ($0A),Y  GETS (FB09...FB10)
2682- 29 DF      1080        AND #$DF     STRIP OFF LOWER CASE BIT
2684- 6D 74 26   1090        ADC $2674    ACCUMULATE SHIFTED SUM
2687- 8D 74 26   1100        STA $2674
268A- 2E 74 26   1110        ROL $2674    SHIFT RESULT, CARRY INTO BIT 0
268D- C8         1120        INY
268E- CC 77 26   1130        CPY $2677    DO IT 8 TIMES
2691- D0 ED      1140        BNE .1
2693- 98         1150        TYA          A = Y = 8
2694- 0A         1160        ASL          FORM $80 BY SHIFTING
2695- 0A         1170        ASL
2696- 0A         1180        ASL
2697- 0A         1190        ASL
2698- A8         1200        TAY          $80 TO Y FOR LATER TRICK
2699- 4D 74 26   1210        EOR $2674    MERGE WITH PREVIOUS "SUM"
269C- 69 0B      1220        ADC #11      FORM $00 FOR VALID ROMS
269E- D0 03      1230        BNE .2       ...NOT A VALID ROM
26A0- A5 0C      1240        LDA $0C      GET MACHINE ID BYTE
26A2- 60         1250        RTS
26A3- A9 00      1260 .2     LDA #0       SIGNAL INVALIDITY
26A5- 60         1270        RTS
```

The pointer at $0A,0B was set up to point to $FB09 using very sneaky code at $248A. Location $2674 initially contains a 0, and $2677 contains an 8. Only the bytes from $FB09 through $FB10 are checksummed. Truthfully, "checksummed" is not the correct word.

The wizards who put ProDOS together figured out a fancy function which changes the 64 bits from $FB09 through $FB10 into the value $75. Their function does this whether your ROMs are the original monitor ROM from 1977-78, the Autostart ROM, the original //e ROM, or any other standard Apple ROM. The values in $FB09-FB10 are not the same in all cases, but the function result is always $75. However, a Franklin ROM does not produce $75. Probably a BASIS also gives a different result, and other clones. Once $75 is obtained, further slippery code changes the value to $00.

S-C Macro Assembler Version 2.0......DOS $100, ProDOS $100, both for $120
Version 2.0 DOS Upgrade Kit for 1.0/1.1/1.2 owners..................$20
ProDOS Upgrade Kit for Version 2.0 DOS owners.......................$30
Source Code of S-C Macro 2.0 (DOS only).................additional $100
Full Screen Editor for S-C Macro (with complete source code)..........$49
S-C Cross Reference Utility......without source code $20, with source $50
RAK-Ware DISASM...$30
Source Code for DISASM...............................additional $30
S-C Word Processor (with complete source code)......................$50
DP18 Source and Object..$50
Double Precision Floating Point for Applesoft (with source code)......$50
ES-CAPE (Extended S-C Applesoft Program Editor)..........
Including Version 2.0 With Source Code..........$50
ES-CAPE Version 2.0 and Source Code Update (for Registered Owners)....$30
"Bag of Tricks", Worth & Lechner, with diskette............($39.95) $36 *
"Bag of Tricks 2"..($49.95) $45 *
MacASM -- Macro Assembler for MacIntosh (Mainstay).........($150.00) $100 *
S-C Documentor (complete commented source code of Applesoft ROMs).....$50
Source Code of //e CX & F8 ROMs on disk.............................$15
Cross Assemblers for owners of S-C Macro Assembler.....$32.50 to $50 each
(Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048,
8051, 8085, 1802/4/5, PDP-11, GI1650/70, others)

AAL Quarterly Disks........................each $15, or any four for $45
Each disk contains the source code from three issues of AAL,
saving you lots of typing and testing.
The quarters are Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.
(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

Diskettes (with hub rings)....................... package of 20 for $20 *
Vinyl disk pages, 6"x8.5", hold two disks each.................10 for $6 *
Diskette Mailing Protectors (hold 1 or 2 disks).............40 cents each
(Cardboard folders designed to fit 6"X9" Envelopes.) or $25 per 100 *
Envelopes for Diskette Mailers.......................... 6 cents each

65802 Microprocessor (Western Design Center)..................($95) $50 *
quikLoader EPROM System (SCRG)...............................($179) $170 *
PROmGRAMER (SCRG)..($149.50) $140 *
Switch-a-Slot (SCRG).......................................($179.50) $170 *
Extend-a-Slot (SCRG)..($35) $32 *

"65816/65802 Assembly Language Programming", Fischer........($19.95) $18 *
"Programming the 65816", Eyes...............................($22.95) $21 *
"Apple //e Reference Manual", Apple Computer................($24.95) $23 *
"Apple //c Reference Manual", Apple Computer................($24.95) $23 *
"ProDOS Technical Reference Manual", Apple Computer.........($29.95) $27 *
"Now That You Know Apple Assembly Language...", Gilder......($19.95) $18 *
"Apple ProDOS: Advanced Features for Programmers", Little..($17.95) $17 *
"Inside the Apple //c", Little..............................($19.95) $18 *
"Inside the Apple //e", Little..............................($19.95) $18 *
"Apple II+/IIe Troubleshooting & Repair Guide", Brenner.....($19.95) $18 *
"Apple][Circuit Description", Gayler......................($22.95) $21 *
"Understanding the Apple II", Sather........................($22.95) $21 *
"Understanding the Apple //e", Sather.......................($24.95) $23 *
"Enhancing Your Apple II, vol. 1", Lancaster................($15.95) $15 *
"Enhancing Your Apple II, vol. 2", Lancaster................($17.95) $17 *
"Assembly Cookbook for the Apple II/IIe", Lancaster.........($21.95) $20 *
"Beneath Apple DOS", Worth & Lechner........................($19.95) $18 *
"Beneath Apple ProDOS", Worth & Lechner.....................($19.95) $18 *
"Real Time Programming -- Neglected Topics", Foster..........($9.95) $9 *
"Microcomputer Graphics", Myers.............................($14.95) $14 *
"Assem. Language for Applesoft Programmers", Finley & Myers.($16.95) $16 *
"Assembly Lines -- the Book", Wagner........................($19.95) $18 *
"AppleVisions", Bishop & Grossberger........................($39.95) $36 *

* On these items add $2.00 for the first item and
$.75 for each additional item for US shipping.
Foreign customers inquire for postage needed.
Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 ***
*** Master Card, VISA, Discover and American Express ***

This means the normal Apple runs a hair slower than the clock rate. But also remember that dynamic RAM needs refreshing from time to time. The refresh of the 256K RAM on the Transwarp card occurs once out of every 16 Apple phase 0 (1MHz) clock cycles. During each 16th 1MHz cycle, the Transwarp slows down to 1MHz. This means that in the time a normal Apple would execute 16 clock cycles, the full-speed Transwarp will execute 53 clock cycles. If not for the long refresh cycle, Transwarp would execute 56 cycles during 16 phase 0 cycles. Now 53 divided by 16 is 3.3125, showing that the maximum speedup factor for Transwarp is 3.3125. I don't know for certain, but the Titan Accelerator II probably has the same characteristic. If so, they both run at a full 3.5 times faster for 15 microseconds, slow down for one microsecond, and then take off again.

The SpeedDemon, on the other hand, can run at a full 3.5 times faster for somewhat longer bursts. If every byte needed is in the SpeedDemon cache memory (static RAM, needing no refresh), execution should proceed at 3.5 times normal Apple speed. Normal programs, however, which are long enough to make us worry about speed, will never be entirely inside the cache. In all comparison tests of real software, Transwarp is faster than either SpeedDemon or Titan. SpeedDemon loses due to its cache, and Titan loses because it does not speed up any accesses to AUXMEM.

The S-C Word Processor increased its speed by about 3.2 for compute-bound operations like searching. Interestingly, an operation that is limited by screen output, like inserting characters from the yank buffer, showed almost no increase in speed. In THE Spreadsheet (MagiCalc) the acceleration factor was about 3.1-3.3, running in a II+ with a Viewmaster 80-column card. Our mailing label system, written mostly in Applesoft, showed a pretty consistent 3.3 speedup. Programs which involve disk I/O will not speed up as much, because the disk still spins at the same 300 rpm.

All in all, we think the Transwarp is a good investment: you get a quality product at a reasonable price which significantly enhances the performance of your computer.

New Book by Tom Weishaar, reviewed by Bob Sander-Cederlof

A little over a year ago, just before he started the "Open-Apple" newsletter, Tom wrote a book. Info Books has just released it, called "Your Best Interest: A Money Book for the Computer Age." It's not about Apple assembly language, but I cannot resist telling you about it anyway!

The book is about interest rates -- how to understand them, how to calculate them, how they affect you. It was written for people who know how to use a spreadsheet program. All the hard math and books of tables are replaced your favorite calc-alike.

```
0845- C8          1350          INY
0846- C0 08       1360          CPY #8
0848- 90 D2       1370          BCC .1
084A- 98          1380          TYA
084B- 0A          1390          ASL
084C- 0A          1400          ASL
084D- 0A          1410          ASL
084E- 0A          1420          ASL
084F- 0D 67 08    1430          ORA X
0852- 20 57 08    1440          JSR B
0855- 69 0B       1450          ADC #$0B
                  1460 *--------------------------------
0857- 48          1470 B        PHA
0858- 08          1480          PHP
0859- 20 DA FD    1490          JSR $FDDA
085C- A9 A0       1500          LDA #" "
085E- 20 ED FD    1510          JSR $FDED
0861- 20 ED FD    1520          JSR $FDED
0864- 28          1530          PLP
0865- 68          1540          PLA
0866- 60          1550          RTS
                  1560 *--------------------------------
0867-             1570 X        .BS 1
                  1580 *--------------------------------
0868- C1 D0 D0
086B- CC C5 A0
086E- DD DB       1590 S1       .AS -/APPLE ][/
0870- B0 A2 20
0873- 4A FF 38
0876- B0 9E       1600 S2       .HS B0.A2.20.4A.FF.38.B0.9E
                  1610 *--------------------------------
0878- 8D 8D       1620 TITLE    .HS 8D8D
087A- CC C4 C1
087D- A0 C1 CE
0880- C4 A0 C1
0883- C4 C3 A0
0886- D3 D4 C1
0889- A0 D2 CF
088C- CC          1630          .AS -/LDA AND ADC STA ROL/
088D- 8D 00       1640          .HS 8D00
                  1650 *--------------------------------
                  1660 PT
088F- A0 00       1670          LDY #0
0891- B9 78 08    1680 .1       LDA TITLE,Y
0894- F0 06       1690          BEQ .2
0896- 20 ED FD    1700          JSR $FDED
0899- C8          1710          INY
089A- D0 F5       1720          BNE .1
089C- 60          1730 .2       RTS
                  1740 *--------------------------------
```

The checksummer can be defeated. The best way, preserving the various side effects, is to change the byte at $269F from $03 to $00. This changes the BNE to an effective no-operation, because it will branch to the next instruction regardless of the status. Another way to get the same result is to store $EA at both $269E and $269F. Still another way is to change the "LDA #0" at $26A3,4 to "LDA $0C" (A5 0C), so that either case gives the same result.

If it thinks it is in a valid Apple computer, the checksummer returns a value in the A-register which is non-zero, obtained from location $0C. The value at $0C has been previously set by looking at other locations in the ROM, trying to tell which version is there. Part of this code is at $2402 and following, and part is at $2047 and following. The byte at $0C will eventually become the Machine ID byte at $BF98 in the System Global Page, so it also gets some bits telling how much RAM is available, and whether an 80-column card and a clock card are found.

Get TransWarp™. The fastest accelerator you can buy for your Apple™ IIe, II, or II+.

Computing at warp speed!

It's an experience you shouldn't miss. And with TransWarp, you won't have to. Because TransWarp will run your software up to 3.6 *times faster* — leaving other accelerators in the stardust!

No more yawning while your Apple™ slowly rearranges text or calculates spreadsheets. With 256K of ultra-fast RAM, TransWarp speeds up *all* Apple software — including AppleWorks, Supercalc 3a, Visicalc, and all educational software, graphics and games. And it's compatible with all standard peripheral cards (such as RamWorks II and Apple memory cards), Profile and Sider hard disks, 3½" UniDisks, 80-column cards, modems, clock cards, mouses and more! You name it, TransWarp accelerates it. There's even a 16 bit upgrade chip available should 16 bit software become available for the Apple.

> *"I recommend Applied Engineering products wholeheartedly."*
>
> *Steve Wozniak, the creator of Apple Computer*

An important difference.

TransWarp's not the only speedup card on the market. But it's the only one that accelerates your Apple's main memory, ROM *and auxiliary memory*. And with more and more programs residing in auxiliary memory, buying anyone else's accelerator makes less and less sense. TransWarp even works with most D.M.A. devices including the Swyft™ card.

There's one more difference. Since TransWarp doesn't use memory caching, it accelerates *all* software — and not just most of it.

A cinch to use.

Simply plug TransWarp into any slot in your Apple II, II+ or IIe including slot 3 in the IIe. Instantly you'll be computing at speeds you only dreamed about before. And should you ever wish to run at normal speed, simply press the ESC key while turning your Apple on.

Since TransWarp is completely transparent, you won't need pre-boot disks or special software. It's ready to go right out of the package!

Speed = Productivity

Imagine the productivity gains you'll achieve when your programs are running over three times faster. TransWarp is so powerful, your Apple will make IBM PCs™ and even ATs™ look like slowpokes — whether you're planning taxes, plotting charts or playing games! Take a look at a few of the features that set TransWarp apart.

- 3.6 MHZ 65C02
- 256K of ultra-fast on-board RAM
- Accelerates main *and* auxiliary memory
- Low power consumption for cool operation
- Totally transparent operation with all software
- Plugs into any slot, including slot 3 on the Apple IIe
- Accelerated 16 bit option available

Satisfaction guaranteed!

Give *your* Apple the TransWarp advantage. With our risk free 15-day money back guarantee, you have nothing to lose but wasted time. Call today!

TransWarp Accelerator
16 bit upgrade (may add later)

For fast response:

Call Applied Engineering, 9 a.m. to 11 p.m. 7 days at (214) 241-6060. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 if outside U.S.A.

Or mail check or money order to Applied Engineering, P.O. Box 798, Carrollton, TX 75006

ÆApplied Engineering
The Apple enhancement experts.

P. O. Box 798, Carrollton, TX 75006 (214) 241-6060

If you have a non-standard Apple or a clone the bytes which are checked to determine which kind of ROM you have may give an illegal result. The following table shows the bytes checked, and the resulting values for $0C. The values in parentheses are not ever checked, but I included them for completeness. The value in $0C will be further modified to indicate the amount of RAM found and the presence of a clock card.

Version	FBB3	FB1E	FBC0	FBBF	$0C
Original Apple II	38	(AD)	(60)	(2F)	00
Autostart, II Plus	EA	AD	(EA)	(EA)	40
Original //e	06	(AD)	EA	(C1)	80
Enhanced //e	06	(AD)	E0	(00)	80
DEBUG //e	06	(AD)	E1	(00)	80
Original //c	06	(4C)	00	FF	88
//c Unidisk 3.5	06	(4C)	00	00	88
/// Emulating II	EA	8A	(??)	(??)	C0

By the way, ProDOS 1.1.1 will not allow booting by an Apple /// emulating a II Plus, possibly because the standard emulator only emulates a 48K machine.

I have no idea what a clone would have in those four locations, but chances are it would be different. You should probably try to fool ProDOS into thinking you are in a II Plus, because most clones are II Plus clones. This means you should somehow change the ID procedures so that the result in $0C is a value of $40. One way to do this is change the code at $2402 and following like this:

```
      Standard                        Change to

2402- A9 00    LDA #0         2402- A9 40    LDA #$40
2404- 85 0C    STA $0C        2404- 4C 2E 24 JMP $242E
2406- A3 B3 FB LDX $FBB3
```

If your clone or modified ROM is a //e, change $2402 to LDA #$80 instead.

You may also need to modify the code at $2047 and following. If you are trying to fool ProDOS into thinking you are an Apple II Plus or //e, and have already made the change described above, change $2047-9 like this:

```
      Standard                     Change to

2047- AE B3 FB LDX $FBB3    2047- 4C 6D 20 JMP $206D
```

No doubt future versions of ProDOS will make provision for clones and modified ROMs even more difficult. And there are always the further problems encountered by usage of the ROMs from BASIC.SYSTEM and the ProDOS Kernel and whatever application program is running.

I am intrigued about seeing what the minimum amount of code is that can distinguish between the four legal varieties of ROM for ProDOS. I notice from the table above that I can identify the four types and weed out the ///emulator by the following simple code at $2402:

```
             LDA $FBB3
             ORA $FB1E
             LDX #3
        .1   CMP TABLE.1,X
             BEQ .2
             DEX
             BPL .1
             SEC
             RTS
     *
     TABLE.1  .HS BD.EF.AF.4E
     TABLE.2  .HS 00.40.80.88
     *
        .2   LDA TABLE.2,X
             JMP $242E
```

With this code installed, all the code from $2047-$206C is not needed, and the JMP $206E should be installed at $2047. The new code at $2402 fits in the existing space with room to spare. Can you do it with even shorter code?

Even Faster 65802 16x16 Multiply......Bob Sander-Cederlof

Bob Boughner, faithful reader from Yorktown, Virginia, decided that the challenge at the end of my article in the January 1986 AAL could not be ignored. He was able to slightly increase the speed of my 16x16 multiply subroutine for the 65802. After studying his code, I made a few more little changes and squeezed out even more cycles.

To see just how much faster the new subroutine is, I carefully counted the cycles, and then went back and did the same to January's subroutine. For some reason I got a new answer for January's program, slightly slower than published. Here are the results:

	Minimum	Maximum	Average
January	333	693	513
New One	321	633	477

The times include 6 cycles for a JSR to call the subroutine, and 6 cycles for the RTS to return. By putting the code in-line, even these 12 cycles could be eliminated. The so-called average time is merely the arithmetic average of the minimum and maximum times. The "real" average for random factors will be faster, because one or both of the INC instructions at lines 1350 and 1430 would be skipped. In fact, almost always at least one would be skipped, saving 48 cycles. Note also that if the factor in CAND is zero, the total time is only 45 cycles.

In counting cycles I assumed that the D-register, which tells the 65802 where the direct page is, has a low byte = 0. If it is non-zero, all of the references to CAND, PLIER, and PROD would require one more cycle.

The new subroutine is only 4 bytes longer than the January one. The new one uses the Y-register, while the old one did not. There are three tricks in the new code which save time. The first one is holding the multiplicand in the Y-register, so that TYA instructions can be used at lines 1310 and 1390. This saves 2 cycles each time, or a total of 32 cycles in the maximum case. The cost is the LDY CAND in line 1200, 4 cycles.

The second trick eliminates the CLC instruction before the multiplier is added in lines 1370-1430. The savings is 16 cycles maximum, and the cost is 8 cycles to set it up in lines 1120-1140 by inverting the high byte of the multiplier. This doesn't affect the average time any, but it does lower the maximum time.

The third trick is at lines 1280 and 1290. I saved 24 cycles by eliminating January's AND ##$0080 instruction here. The LDA PLIER-1 instruction picks up the low byte of the multiplier in the high byte of the A-register, allowing me to see what bit 7 of the multiplier is without any masking or shifting.

```
                           .OP 65802
                   *SAVE BOUGHNER'S MULTIPLY
                   *--------------------------------
                   *   CONTRIBUTED BY BOB BOUGHNER
                   *   MODIFIED A LITTLE MORE BY BOB S-C
                   *--------------------------------
00-                CAND    .EQ 0,1
02-                PLIER   .EQ 2,3
04-                PROD    .EQ 4,5,6,7
                   *--------------------------------
                   MUL.FASTER.YET.16X16.65802
000800- A2 08              LDX #8        WILL LOOP 8 TIMES
000802- A5 03              LDA PLIER+1   INVERT HIGH BYTE
000804- 49 FF              EOR #$FF      TO SAVE "CLC" IN LOOP
000806- 85 03              STA PLIER+1
000808- 18                 CLC
000809- FB                 XCE           ENTER "NATIVE" MODE
00080A- C2 30              REP #$30      16-BITS BOTH X & M
00080C- 64 04              STZ PROD      CLEAR PRODUCT
00080E- 64 06              STZ PROD+2
000810- A4 00              LDY CAND      MULTIPLICAND IN Y-REG
000812- D0 06              BNE .2        ...NON-ZERO, START LOOP
000814- FB                 XCE           ...ZERO, EXIT NOW
000815- 60                 RTS
                   *--------------------------------
000816- 06 04      .1      ASL PROD      DOUBLE THE PRODUCT
000818- 26 06              ROL PROD+2
                   *--------------------------------
00081A- A5 01      .2      LDA PLIER-1   GET LOW BYTE IN A(15-8)
00081C- 10 0A              BPL .3        ...ORIG. BIT=0, DON'T ADD
00081E- 18                 CLC
00081F- 98                 TYA           ...ORIG. BIT=1, ADD 'CAND
000820- 65 04              ADC PROD
000822- 85 04              STA PROD
000824- 90 02              BCC .3
000826- E6 06              INC PROD+2    ADD CARRY TO HI-16
                   *--------------------------------
000828- 06 02      .3      ASL PLIER     SHIFT MULTIPIER, GET HI-BIT
00082A- B0 09              BCS .4        ...ORIG. BIT=0, DON'T ADD
00082C- 98                 TYA           ...ORIG. BIT=1, ADD 'CAND
00082D- 65 05              ADC PROD+1    ADD TO MIDDLE OF PRODUCT
00082F- 85 05              STA PROD+1
000831- 90 02              BCC .4
000833- E6 07              INC PROD+3    (NEVER BOTHERS PROD+4)
                   *--------------------------------
000835- CA         .4      DEX
000836- D0 DE              BNE .1
000838- 38                 SEC
000839- FB                 XCE
00083A- 60                 RTS
                   *--------------------------------
```

Some More Rumors

Electronics magazine printed a brief news item about a second source for 65816 chips. Western Design has signed up a lot of licensees to make these chips, but none of them are in production as of this month. Electronics says VLSI Technology Inc., of San Jose, California, is projecting prices in the $10 range for volume purchases. When? Target is to start selling sample quantities next summer. Meanwhile, volume prices are in the $35 range from Western Design Center. The single-unit price is still about $100.

The parts we are selling are the 65C802 from Western Design Center. Our price to you is $50 each. These are normally spec'd at 2MHz, but sometimes we get 4MHz parts at the same price, when they are out of the slower ones. Either speed works equally well in an Apple motherboard, but you need the 4MHz chip to use in a Transwarp accelerator card.

Continued on page 18

Add Smarts to 65816 Dis-Assembler..............Jim Poponoe

I found fascinating the article by Bob Sander-Cederlof in the March 1985 AAL, entitled "A Disassembler for the 65816". I purchased AAL Quarterly Disk 18 and tried it out for myself, watching 65802 instructions zip before my eyes.

But, whoa! Bob was correct in warning that his disassembler would not know whether immediate-mode instructions are two or three bytes long. Bob explained "only by executing the programming, and tracing it line-by-line, can we tell." A fully accurate disassembler for the 65816 would have to execute the equivalent of STEP and TRACE, following the logic flow of the program.

I wanted an easier, quick-and-dirty way to spiff up the output, one that would at least recognize simple, straightforward changes in the processor status. I reasoned that:

1) Interpretation of immediate-mode instructions depends on the state of E, M, and X bits in the status register.

2) E and C bits are exchangeable.

3) The disassembler must keep track of all four bits (C, E, X, and M) in order to disassemble immediate mode opcodes correctly.

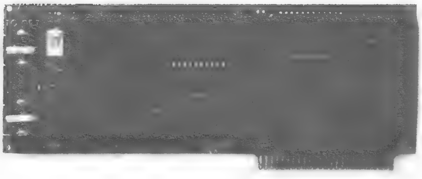
4) The disassembler should also keep track of when the processor status is pushed onto or pulled off the stack.

My implementation assigns a memory location for the E-bit, and a small "stack" of 8 memory locations for the status register. One more memory location serves as the stack pointer. Here is the initialization code for these memory locations, replacing lines 1450-1480 in Bob's March 1985 listing:

```
000810- A2 FF       1450        LDX #$FF          START WITH E=1
000812- 8E 1D 08    1454        STX E.BIT
000815- 8E 1F 08    1458        STX STATUS.STACK  EMPTY THE STATUS STACK
000818- E8          1462        INX               X=0
000819- 8E 1E 08    1466        STX STATUS.PNTR
00081C- 60          1470        RTS
                    1474 *-------------------------------
00081D-             1478 E.BIT          .BS 1
00081E-             1482 STATUS.PNTR    .BS 1
00081F-             1486 STATUS.STACK   .BS 8
```

I added a JSR TEST.OP.CODES line at 5865, to call some new code which looks for CLC, SEC, REP, SEP, PHP, PLP, and XCE instructions. It adjusts the flags appropriately in response to these instructions. If the current opcode is none of the above, TEST.OP.CODES checks the status bits and the opcode to set up the correct immediate-mode length. If the opcode is an immediate mode operation on the A-register, and if E=0 and M=0, then 16-bit immediate will be disassembled. If the opcode is an immediate mode operation on the X- or Y-register, and if E=0 and X=0, then 16-bit immediate will be disassembled. Otherwise, any kind of immediate mode instruction will be

Viewmaster 80,™ the sharpest 80 column card of them all.



	BUILT-IN SOFTSWITCH	SHIFT KEY SUPPORT	LOW POWER DESIGN	80 COLUMN HOME	7 × 9 DOT MATRIX	LIGHT PEN INPUTS	40 COLUMN OVERRIDE	INVERSE CHARACTERS	
VIEWMASTER 80	✓	✓	✓	✓	✓	✓	✓	✓	
SUPRTERM		✓					✓	✓	
WIZARD 80					✓		✓	✓	
VISION 80	✓	✓			✓				
OMNIVISION		✓					✓	✓	
VIEWMAX 80	✓	✓			✓			✓	
SMARTERM	✓	✓				✓	✓		
VIDEOTERM		✓	✓		✓	✓		✓	

Now, get great resolution and total software compatibility for your Apple II® or II+®.

One look at the chart will give you some of the reasons there's only one smart choice in 80 column cards for your Apple. But the real secret to Viewmaster 80's success is something even better: Total compatibility.

The Viewmaster 80 works with all 80 column applications, including DOS 3.3, PRO-DOS, CP/M, Pascal, WordStar, Format II, Easywriter, Applewriter II, Supertext 80, Zardax, Apple PI, Letter Perfect, dBASE II, Visicalc, Multiplan, and *hundreds* of others.

And the Viewmaster 80 delivers a super sharp, state-of-the-art display with a 7 × 9 character matrix for clear, easily readable characters. Here are just a few of the powerful features the Viewmaster 80 delivers for a great price ($139):

• 80 Characters by 24 lines • Fully compatible with all Apple languages and software • Super sharp 7 × 9 character matrix with true descenders • Highest compatibility with existing 80 column software • Power and input connector for light pen • Very low power consumption • High speed (18 MHZ) scroll rate • Upper and lower case characters with true descenders, both inverse and normal; all on-screen editing functions are supported • User-definable cursor shape • Compatible with Apple II, II+ and IIe • Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Æ Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

disassembled with an 8-bit operand.

I tried the program on all the sample 65802 code I could find, and it was all disassembled correctly. Of course it is certainly possible to fool my program. The C-bit, and hence possibly the E-bit, can be changed in many other ways than by using the CLC and SEC instructions. The program flow is not followed, so it is possible than my emulation of the carry status and the XCE will not agree with what happens in some code. If you adhere to the "nice" standard of always using explicit SEC or CLC opcodes before an XCE opcode, the disassembler should stay in step perfectly.

When you type 800G to link in the disassembler (refer to Bob's article to know what I mean here) the status is initialized to E=C=M=X=1. This means normal 6502 mode. If you disassemble some code with XCE's in it, the status I keep will probably be left in some other mode. If you then try to disassemble some plain vanilla 6502 code, the immediate instructions may be disassembled with 16-bit operands. Just type 800G again to get back to normal.

By the way, in working with Bob's disassembler I discovered a typing error in his code. Line 3980 was originally >OXA TAY, and it should have been >OXA DEY. The hex listing in Bob's article showed $AF stored in $963; it really should be $89. Without this change, the DEY opcode disassembles as TAY!

The listing that follows has been extensively modified by Bob, based on my code I sent him last September. The lines are numbered to follow after the last line of the program on the quarterly disk.

```
                       7060 *--------------------------------
                       7070 TEST.OP.CODES
000C1A- 48             7080        PHA         SAVE OPCODE
000C1B- 46 00          7090        LSR IMM.SIZE     ASSUME 8-BIT IMMEDIATE
000C1D- AE 1E 08       7100        LDX STATUS.PNTR
000C20- C9 18          7110        CMP #$18    CLC?
000C22- F0 36          7120        BEQ CLC.OP
000C24- C9 38          7130        CMP #$38    SEC?
000C26- F0 3C          7140        BEQ SEC.OP
000C28- C8             7150        INY
000C29- C9 C2          7160        CMP #$C2    REP?
000C2B- F0 3E          7170        BEQ REP.OP
000C2D- C9 E2          7180        CMP #$E2    SEP?
000C2F- F0 44          7190        BEQ SEP.OP
000C31- 88             7200        DEY
000C32- C9 08          7210        CMP #$08    PHP?
000C34- F0 47          7220        BEQ PHP.OP
000C36- C9 28          7230        CMP #$28    PLP?
000C38- F0 53          7240        BEQ PLP.OP
000C3A- C9 FB          7250        CMP #$FB    XCE?
000C3C- F0 56          7260        BEQ XCE.OP
                       7270 *--------------------------------
000C3E- 29 1F          7280        AND #$1F    ORA, AND, EOR, ADC, BIT, LDA,
000C40- C9 09          7290        CMP #$09                         CMP, SBC?
000C42- 08             7300        PHP         SAVE ANSWER
000C43- A9 20          7310        LDA #$20    ASSUME M-BIT
000C45- 28             7320        PLP         GET PREVIOUS ANSWER
000C46- F0 01          7330        BEQ .1      IT IS M-BIT
000C48- 4A             7340        LSR   (LDA #$10)    USE X-BIT INSTEAD
000C49- 3D 1F 08       7350 .1     AND STATUS.STACK,X
000C4C- D0 0A          7360        BNE .2      ...USE 8-BIT IMMEDIATE
000C4E- AD 1D 08       7370        LDA E.BIT
000C51- 4A             7380        LSR
```

```
000C52- B0 04      7390          BCS .2        E=1, USE 8-BIT IMMEDIATE
000C54- A9 FF      7400          LDA #$FF      ...USE 16-BIT IMMEDIATE
000C56- 85 00      7410          STA IMM.SIZE
000C58- 68         7420 .2       PLA           GET OPCODE AGAIN
000C59- 60         7430          RTS
                   7440 *--------------------------------
000C5A- BD 1F 08   7450 CLC.OP LDA STATUS.STACK,X
000C5D- 29 FE      7460          AND #$FE
                   7470 UPDATE.STATUS
000C5F- 9D 1F 08   7480          STA STATUS.STACK,X
000C62- 68         7490          PLA
000C63- 60         7500          RTS
                   7510 *--------------------------------
000C64- BD 1F 08   7520 SEC.OP LDA STATUS.STACK,X
000C67- 09 01      7530          ORA #$01
000C69- D0 F4      7540          BNE UPDATE.STATUS   ...ALWAYS
                   7550 *--------------------------------
000C6B- B1 3A      7560 REP.OP LDA (PCL),Y      LOOK AT OPERAND
000C6D- 49 FF      7570          EOR #$FF
000C6F- 3D 1F 08   7580          AND STATUS.STACK,X
000C72- 4C 5F 0C   7590          JMP UPDATE.STATUS
                   7600 *--------------------------------
000C75- B1 3A      7610 SEP.OP LDA (PCL),Y
000C77- 1D 1F 08   7620          ORA STATUS.STACK,X
000C7A- 4C 5F 0C   7630          JMP UPDATE.STATUS
                   7640 *--------------------------------
000C7D- BD 1F 08   7650 PHP.OP LDA STATUS.STACK,X
000C80- E8         7660          INX
000C81- E0 08      7670          CPX #8
000C83- 90 02      7680          BCC PHP.PLP
000C85- A2 00      7690          LDX #0
                   7700 PHP.PLP
000C87- 8E 1E 08   7710          STX STATUS.PNTR
000C8A- 4C 5F 0C   7720          JMP UPDATE.STATUS
                   7730 *--------------------------------
000C8D- CA         7740 PLP.OP DEX
000C8E- 10 F7      7750          BPL PHP.PLP
000C90- A2 07      7760          LDX #7
000C92- F0 F3      7770          BEQ PHP.PLP
                   7780 *--------------------------------
000C94- 4E 1D 08   7790 XCE.OP LSR E.BIT    GET E-BIT INTO CARRY
000C97- 08         7800          PHP          SAVE IT
000C98- BD 1F 08   7810          LDA STATUS.STACK,X
000C9B- 8D 1D 08   7820          STA E.BIT    NEW E-BIT
000C9E- 4A         7830          LSR          C-BIT INTO CARRY
000C9F- 90 02      7840          BCC .1       ...NEW E-BIT = 0
000CA1- 09 18      7850          ORA #$18     ...NEW E-BIT=1, SO SET M=X=1
000CA3- 28         7860 .1       PLP          GET NEW C-BIT (OLD E-BIT)
000CA4- 2A         7870          ROL          PUT IT INTO STATUS BYTE
000CA5- 4C 5F 0C   7880          JMP UPDATE.STATUS
```

Further notes by Bob Sander-Cederlof:

Thanks, Jim! Your ideas were a big help! In looking back over my work, I noticed some more improvements.

R. F. O'Brien wrote us just this week with the news that he had found two bugs in the disassembler. One was the typing error at line 3980 which Jim noted above. But Robert found a second typo, at line 4960. ">OXB LDX" should be changed to ">OXB CPX". This changes the byte shown in the original article at $9BF from $19 to $0F.

I found a way to simplify the >ON macro, which speeds up assembly and shortens the listing. Replace lines 1220-1290 with the following:

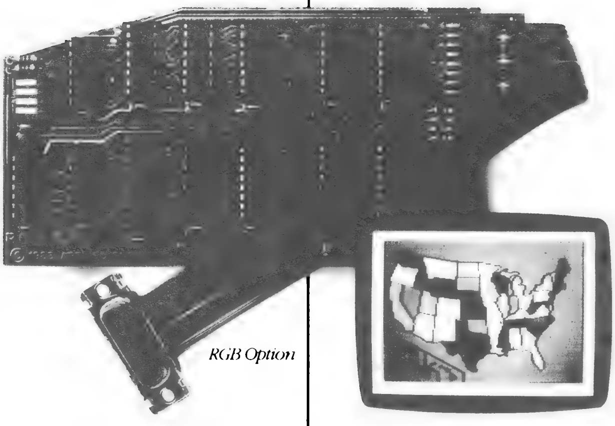
```
1220          .MA ON
1280 ]1]2]3]4 .DA ']1-64*32+']2-64*32+']3-64*2
1290          .EM
```

of RamWorks II memory. And speaking of more memory, RamWorks II has a memory expansion connector on board so a low profile (no slot 1 interference) memory expansion card can add another 512K or 2 MEG of memory.

Unlike Apple's smaller, more expensive RAM card, RamWorks II plugs into the IIe auxiliary slot and therefore leaves slots 4 and 5 available for other peripheral cards.

It's In Color

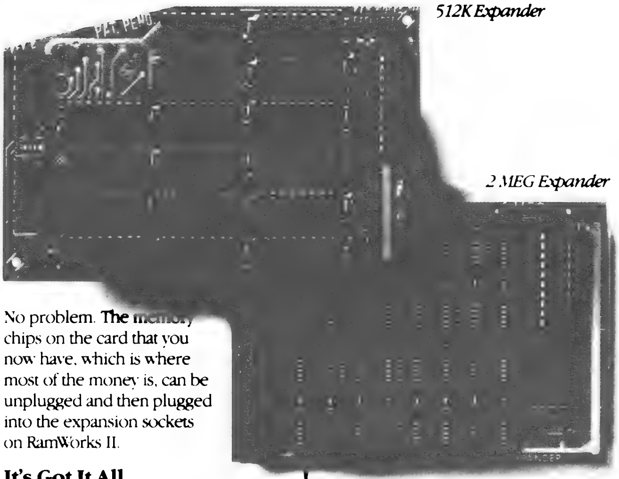
RamWorks II by itself is *fully* compatible with both the Apple monochrome and color monitors. But if you want better color graphics *plus* a more readable 80 column text (that blows away any composite color monitor) you'll appreciate our RGB color option. For only $129, it can be added to RamWorks II, giving you a razor sharp, vivid brilliance that's unsurpassed in the industry. The RGB option does not waste another valuable slot, but rather plugs into the back of RamWorks II with no slot 1 interference (works on the original RamWorks, too) and attaches to virtually any RGB monitor. And remember. You can order the RGB option with your RamWorks II. Or add it on at a later date.



RGB Option

It Corrects Mistakes.

Let's say you bought some other RAM card (and that's a mistake) and your RAM card is not being recognized by AppleWorks, Advanced Visicalc, Flashcalc, Supercalc 3A, or other programs, and you want RamWorks II. No problem. The memory chips on the card that you now have, which is where most of the money is, can be unplugged and then plugged into the expansion sockets on RamWorks II.



512K Expander

2 MEG Expander

It's Got It All.

- 15 Day Money Back Guarantee
- Super sharp 80 column text (patent pending) with or without RGB option
- Double high resolution graphics (with or without RGB option)
- Expandable up to 1 Meg (1024K) on main board
- Expandable to 3 Meg (3072K) with expander (piggyback) card
- Can use 64K or 256K RAMS in any combination
- Linear addressing coprocessor port
- Automatic AppleWorks expansion up to 2277K desktop
- Accelerates AppleWorks
- Built-in AppleWorks printer buffer
- The only large RAM card that's 100% compatible with all IIe software
- RamDrive™ the ultimate disk emulation software included free
- 16 Bit option
- Compatible RGB option
- Built-in self diagnostics software
- No slot 1 interference
- Lowest power consumption (patent pending)
- Takes only one slot (auxiliary)
- Software industry standard
- Advanced Computer Aided Design
- Used by Apple Computer, Steve Wozniak and virtually all software companies
- Displays date and time on the AppleWorks screen with any PRO-DOS compatible clock
- 5 Year no hassle warranty

RamWorks II with 64K	**$ 179**
RamWorks II with 256K	**$ 219**
RamWorks II with 512K	**$ 269**
RamWorks II with 1 MEG	**$ 389**
RamWorks II with 1.5 MEG	**$ 549**
RamWorks II with 3 MEG	**$1699**
RGB Option (may add later)	**$ 129**
16 Bit Option (may add later)	**$ 89**

RamWorks II. The industry standard for memory expansion of the Apple IIe.

ORDER YOUR RamWorks II TODAY. 9 a.m. to 11 p.m. 7 Days, or send check or money order to Applied Engineering.

MasterCard, Visa and C.O.D. welcome. Texas Residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

I also discovered that one kind of Apple monitor ROM did not have the RELADR subroutine, so I re-coded lines 6760-6950. Replace those lines with the following:

```
                        6760 *---8- OR 16-BIT RELATIVE--------
 000BEB- B1 3A          6770        LDA (PCL),Y  8=OFFSET, 16=OFFSETHI
 000BED- 88             6780        DEY          TEST LENGTH
 000BEE- 84 30          6790        STY FORMATH  =0 IF 8-BIT
 000BF0- F0 04          6800        BEQ .10      ...8-BIT
 000BF2- 85 30          6810        STA FORMATH  ...16-BIT
 000BF4- B1 3A          6820        LDA (PCL),Y  LOW BYTE OF 16-BIT OFFSET
 000BF6- 85 2E          6830 .10    STA FORMATL
 000BF8- 20 53 F9       6840        JSR PCADJ
 000BFB- 18             6850        CLC
 000BFC- 65 2E          6860        ADC FORMATL
 000BFE- AA             6870        TAX
 000BFF- 98             6880        TYA
 000C00- 65 30          6890        ADC FORMATH
 000C02- 4C 41 F9       6900        JMP PRNTAX
```

One last item. I wrote a test routine to call the disassembler for every possible opcode from 00 to FF. Here it is:

```
 000CA8- A0 00          7900 TT     LDY #0
 000CAA- A9 C0          7910        LDA #$C0
 000CAC- 85 3A          7920        STA PCL
 000CAE- A9 02          7930        LDA #2       $2C0...$3C3
 000CB0- 85 3B          7940        STA PCH
 000CB2- 98             7950 .1     TYA
 000CB3- 99 C0 02       7960        STA $2C0,Y
 000CB6- C8             7970        INY
 000CB7- D0 F9          7980        BNE .1
 000CB9- 8C C0 03       7990        STY $3C0
 000CBC- C8             8000        INY
 000CBD- 8C C1 03       8010        STY $3C1
 000CC0- C8             8020        INY
 000CC1- 8C C2 03       8030        STY $3C2
 000CC4- 20 7A 0B       8040 .2     JSR INSTDSP
 000CC7- A0 00          8050        LDY #0
 000CC9- B1 3A          8060        LDA (PCL),Y
 000CCB- C9 FF          8070        CMP #$FF
 000CCD- F0 10          8080        BEQ .3
 000CCF- AD 00 C0       8090 .4     LDA $C000
 000CD2- 10 FB          8100        BPL .4
 000CD4- 8D 10 C0       8110        STA $C010
 000CD7- E6 3A          8120        INC PCL
 000CD9- D0 E9          8130        BNE .2
 000CDB- E6 3B          8140        INC PCH
 000CDD- D0 E5          8150        BNE .2       ...ALWAYS
 000CDF- 60             8160 .3     RTS
```

Continued from page 10

Rumors continue to ricochet around the club newsletter circuit about the possible configuration of the new Apple II (usually called the //x). Most rumor sources agree now that the //x will use a 65C816. We sure HOPE so! One source said he looks for an 8MHz clock. We doubt that, because current projections are for 8MHz chips becoming available about 1st quarter 1987. And the RAM for 8MHz operation would be far too expensive. My guess we will see either 2MHz or 3.58MHz.

Most are now including a SCSI port in their list of features, since the Macintosh Plus has one. Some are talking about a smaller set of normal slots, supplemented by some new super-slots having more signals available. There are reportedly a number of different versions of the //x already in existence, seeded around. If that is true, it could be than no one (even inside Apple) yet knows what the REAL //x will be.

Fastest 6502 Multiplication Yet................Charles Putney
Shankill, Dublin, Ireland

Here is an 8x8 multiply routine that will blow your socks off! The maximum time, including both a calling JSR and a returning RTS, is only 66 cycles! The minimum is 60 cycles, and most factors will multiply in 63 cycles. Recall that the fastest time in Bob S-C's January 1986 AAL article for a 6502 was 132 cycles. My new one is twice as fast!

As with most fast routines, there is a trade off in memory space. My program uses 1024 bytes of lookup tables. This isn't so bad if you really need or want a 2:1 speed advantage.

My routine is based on the fact that:

    4 * X * Y = (X+Y)^2 - (X-Y)^2

I got this idea from an article in EDN Magazine by Arch D. Robison (October 13, 1983, pages 263-4). His routine used the fact that:

    2 * X * Y = X^2 + Y^2 - (X-Y)^2

Robison's method requires three dips into the lookup tables. Formulated to the same method for passing parameters, his method takes either 74 or 77 cycles. Here is my rendition of his method:

```
                  1000 *SAVE ROBISON'S FAST 8X8
                  1010 *--------------------------------
                  1020 *   MODIFIED FROM ORIGINAL PROGRAM
                  1030 *   BY ARCH D. ROBISON, BURROUGHS CORP.
                  1040 *      EDN, OCTOBER 13, 1983.
                  1050 *--------------------------------
                  1060 *   ENTER WITH (A)=MULTIPLIER # 1
                  1070 *              (X)=MULTIPLIER #2
                  1080 *   EXIT WITH (A)=PRODUCT HI BYTE
                  1090 *             (X)=PRODUCT LO BYTE
                  1100 *--------------------------------
06-               1110 PROD    .EQ $06       PRODUCT TEMP OF M1*M2 (LOW BYTE)
07-               1120 M2      .EQ $07       TEMP FOR M2 SAVE
                  1130 *--------------------------------
0800- A8          1140 MULT8   TAY           SAVE M1 IN Y
0801- 86 07       1150         STX M2        SAVE M2
0803- 25 07       1160         AND M2        CHECK IF BOTH FACTORS ARE ODD
0805- 4A          1170         LSR           SET CARRY <--> BOTH ODD
0806- BD 00 09    1180         LDA SQL,X     ADD (X*X)/2 AND (Y*Y)/2
0809- 79 00 09    1190         ADC SQL,Y
080C- 85 06       1200         STA PROD      SAVE LO BYTE OF PRODUCT
080E- BD 00 0A    1210         LDA SQH,X
0811- 79 00 0A    1220         ADC SQH,Y
0814- AA          1230         TAX           SAVE HI BYTE OF PRODUCT
0815- 98          1240         TYA           GET M1 BACK
0816- 38          1250         SEC
0817- E5 07       1260         SBC M2        FIND M1 - M2
0819- B0 04       1270         BCS .1        M1 >= M2, CONTINUE
081B- E9 00       1280         SBC #0        M1 < M2, FORM 2'S COMPLEMENT
081D- 49 FF       1290         EOR #$FF
081F- A8          1300 .1      TAY           USE ABS(M1-M2) AS INDEX
0820- A5 06       1310         LDA PROD        TO FIND SQUARE/2 IN TABLE
0822- F9 00 09    1320         SBC SQL,Y     NOW SUBTRACT (X-Y)*(X-Y)
0825- 85 06       1330         STA PROD      SAVE LO BYTE OF RESULT
0827- 8A          1340         TXA           HI BYTE FROM PREVIOUS SUM
0828- F9 00 0A    1350         SBC SQH,Y
082B- A6 06       1360         LDX PROD      LO BYTE OF FINAL PRODUCT
082D- 60          1370         RTS
```

The entries in the two tables (SQL and SQH) are the squares of the numbers from 0 to 255, divided by two. The low bytes are in the SQL table, and the high bytes are in SQH. Dividing by two throws away an important bit for odd factors, but lines 1160-1170 compensate for the loss.

I looked for a way to add fewer table entries together and came upon the sum^2 - diff^2. Since the sum can be as large as 255+255=510, I need twice as much table space. Lest you despair of typing in such a large table, let me offer an Applesoft program which will write a text file of the source code for the table:

```
100 D$ =  CHR$ (4)
110  PRINT D$"OPEN TEMPFILE"
120  PRINT D$"WRITE TEMPFILE"
1000  REM  CREATE SQUARE/4 TABLE
1010  PRINT "1000 SQL":A$ = "#":L = 1010
1020  FOR I = 0 TO 510 STEP 8: GOSUB 2000
1030  NEXT I
1100  PRINT "2000 SQH":A$ = "/":L = 2010
1110  FOR I = 0 TO 510 STEP 8: GOSUB 2000
1120  NEXT I
1130  PRINT D$"CLOSE": END
2000  REM  GENERATE 8 ITEMS
2010 N =  INT (I * I / 4): PRINT L"  .DA "A$;N;
2020  FOR J = I + 1 TO I + 7
2030 N =  INT (J * J / 4): PRINT ","A$;N;
2040  NEXT J:L = L + 10
2050  PRINT : RETURN
```

My tables contain the squares divided by four. I can hear you saying, "Wait a minute! You can't just divide by four and truncate!" Well, even squares are all multiples of four; odd squares are all multiples of four with a remainder = 1. The sum of two numbers and the difference of the same numbers are either both even or both odd. Therefore, we never lose anything by throwing away our truncated 1.

The number of cycles my MULT8 takes depends on the values of the two factors. You call MULT8 with one factor in the A-register and the other in the X-register. If (A) is less than (X), it takes an extra 3 cycles to perform a complement operation. If the sum of the factors is greater than 255, add another three cycles. To summarize,

	A>=X	A<X
sum<256	60	63
sum>255	63	66

Just for fun, I also wrote a program to generate the square/4 tables. This takes less time than loading the tables from disk, so it could mean faster booting for some hi-resolution game program that needs super-fast multiplications. It is in lines 1560-2100 below.

The origin I used in my program is meant just to allow me to test it. I wrote an Applesoft program to call TEST at $6000 (CALL 24576). The program POKEd two factors at $FA and $FB, called TEST, and then checked the result at the same two

locations. If you want to use MULT8, you should just assemble it along with the rest of your program, without any special origin. You should make sure that the tables start on an even page boundary, or it will cost you up to 8 cycles extra for indexing across a page boundary.

```
                  1000 *SAVE PUTNEY'S FASTER 8X8
                  1010 *-------------------------------
                  1020 *       ULTRA-FAST 8 X 8 MULTIPLY
                  1030 *-------------------------------
                  1040 *    ENTER WITH (A)=MULTIPLIER # 1
                  1050 *               (X)=MULTIPLIER #2
                  1060 *    EXIT WITH (A)=PRODUCT HI BYTE
                  1070 *              (X)=PRODUCT LO BYTE
                  1080 *-------------------------------
                  1090 *    TIMING DATA
                  1100 *       MINIMUM TIME = 54 CYCLES
                  1110 *       MAXIMUM TIME = 60 CYCLES
                  1120 *       AVERAGE TIME = 57 CYCLES
                  1130 *-------------------------------
06-               1140 PROD    .EQ $06       PRODUCT TEMP OF M1*M2 (LOW BYTE)
07-               1150 M2      .EQ $07       TEMP FOR M2 SAVE
                  1160 *-------------------------------
                  1170         .OR $6000     SAFE PLACE
                  1180 *-------------------------------
                  1190 *    TEST FOR APPLESOFT DRIVER
                  1200 *-------------------------------
6000- A5 FA       1210 TEST    LDA $FA       LOAD ACC AND X SO BASIC CAN TEST
6002- A6 FB       1220         LDX $FB
6004- 20 0C 60    1230         JSR MULT8
6007- 86 FA       1240         STX $FA       NOW BASIC CAN CHECK ACC AND X
6009- 85 FB       1250         STA $FB
600B- 60          1260         RTS
                  1270 *-------------------------------
600C- A8          1280 MULT8   TAY           SAVE M1 IN Y
600D- 86 07       1290         STX M2        SAVE M2
600F- 38          1300         SEC           SET CARRY FOR SUBTRACT
6010- E5 07       1310         SBC M2        FIND DIFFERENCE
6012- B0 04       1320         BCS .1        WAS M1 > M2 ?
6014- 49 FF       1330         EOR #$FF      INVERT IT
6016- 69 01       1340         ADC #$01      AND ADD 1
6018- AA          1350 .1      TAX           USE ABS(M1-M2) AS INDEX
6019- 18          1360         CLC
601A- 98          1370         TYA           GET M1 BACK
601B- 65 07       1380         ADC M2        FIND M1 + M2
601D- A8          1390         TAY           USE M1+M2 AS INDEX
601E- 90 11       1400         BCC .2        M1+M2 < 255 ?
6020- B9 00 62    1410         LDA SQL+256,Y      FIND SUM SQUARED LOW IF > 255
6023- FD 00 61    1420         SBC SQL,X          SUBTRACT DIFF SQUARED
6026- 85 06       1430         STA PROD           SAVE IN PRODUCT
6028- B9 00 64    1440         LDA SQH+256,Y      HI BYTE
602B- FD 00 63    1450         SBC SQH,X
602E- A6 06       1460         LDX PROD      GET PROD LOW IN X
6030- 60          1470         RTS           DONE
6031- 38          1480 .2      SEC           SET CARRY FOR SUBTRACT
6032- B9 00 61    1490         LDA SQL,Y     FIND SUM OF SQUARES LOW IF < 255
6035- FD 00 61    1500         SBC SQL,X     SUBTRACT DIFF SQUARED
6038- 85 06       1510         STA PROD      SAVE IN PRODUCT
603A- B9 00 63    1520         LDA SQH,Y     HI BYTE
603D- FD 00 63    1530         SBC SQH,X
6040- A6 06       1540         LDX PROD      GET PROD LOW IN X
6042- 60          1550         RTS
                  1560 *-------------------------------
                  1570 *    PROGRAM TO CREATE A TABLE OF SQUARES/4
                  1580 *-------------------------------
00-               1590 LOTP    .EQ 0,1
02-               1600 HITP    .EQ 2,3
                  1610 *-------------------------------
6043- A0 00       1620 SQUARE LDY #0
6045- 84 00       1630         STY LOTP
6047- 84 02       1640         STY HITP
6049- 8C AE 60    1650         STY SQ
604C- 8C AF 60    1660         STY SQ+1
604F- 8C B0 60    1670         STY SQ+2
```

```
6052- 8C AC 60  1680        STY DELTA+1
6055- 8C AD 60  1690        STY DELTA+2
6058- 8C 00 68  1700        STY $6800
605B- 8C 00 6A  1710        STY $6A00
605E- C8        1720        INY
605F- A9 40     1730        LDA #$40
6061- 8D AB 60  1740        STA DELTA
6064- A9 68     1750        LDA /$6800
6066- 85 01     1760        STA LOTP+1
6068- A9 6A     1770        LDA /$6A00
606A- 85 03     1780        STA HITP+1
606C- A2 01     1790        LDX #1
                1800 *--------------------------------
606E- 18        1810 .1     CLC
606F- AD AB 60  1820        LDA DELTA
6072- 6D AE 60  1830        ADC SQ
6075- 8D AE 60  1840        STA SQ
6078- AD AC 60  1850        LDA DELTA+1
607B- 6D AF 60  1860        ADC SQ+1
607E- 8D AF 60  1870        STA SQ+1
6081- 91 00     1880        STA (LOTP),Y
6083- AD AD 60  1890        LDA DELTA+2
6086- 6D B0 60  1900        ADC SQ+2
6089- 8D B0 60  1910        STA SQ+2
608C- 91 02     1920        STA (HITP),Y
                1930 *--------------------------------
608E- AD AB 60  1940        LDA DELTA
6091- 69 80     1950        ADC #$80
6093- 8D AB 60  1960        STA DELTA
6096- 90 08     1970        BCC .2
6098- EE AC 60  1980        INC DELTA+1
609B- D0 03     1990        BNE .2
609D- EE AD 60  2000        INC DELTA+2
60A0- C8        2010 .2     INY
60A1- D0 CB     2020        BNE .1
60A3- E6 01     2030        INC LOTP+1
60A5- E6 03     2040        INC HITP+1
60A7- CA        2050        DEX
60A8- 10 C4     2060        BPL .1
60AA- 60        2070        RTS
                2080 *--------------------------------
60AB-           2090 DELTA  .BS 3
60AE-           2100 SQ     .BS 3
                2110 *--------------------------------
                2120 *  TABLE OF SQUARES/4 FROM 0 TO 511
                2130 *--------------------------------
60B1-           2140        .BS *+$FF/$100*$100-*  KEEP TABLES ALIGNED ON
                2150 *--------------------------------          PAGE BOUNDARY
6100- 00 00 01
6103- 02 04 06
6106- 09 0C     2160 SQL    .DA #0,#0,#1,#2,#4,#6,#9,#12
6108- 10 14 19
610B- 1E 24 2A
610E- 31 38     2170        .DA #16,#20,#25,#30,#36,#42,#49,#56
6110- 40 48 51
6113- 5A 64 6E
6116- 79 84     2180        .DA #64,#72,#81,#90,#100,#110,#121,#132
6118- 90 9C A9
611B- B6 C4 D2
611E- E1 F0     2190        .DA #144,#156,#169,#182,#196,#210,#225,#240

6300- 00 00 00
6303- 00 00 00
6306- 00 00     2830 SQH    .DA /0,/0,/1,/2,/4,/6,/9,/12
6308- 00 00 00
630B- 00 00 00
630E- 00 00     2840        .DA /16,/20,/25,/30,/36,/42,/49,/56
6310- 00 00 00
6313- 00 00 00
6316- 00 00     2850        .DA /64,/72,/81,/90,/100,/110,/121,/132

64F0- F0 F1 F2
64F3- F3 F4 F5
64F6- F6 F7     3470       .DA /61504,/61752,/62001,/62250,/62500,/62750,
64F8- F8 F9 FA                                               /63001,/63252
64FB- FB FC FD
64FE- FE FF     3480       .DA /63504,/63756,/64009,/64262,/64516,/64770,
                3490 *--------------------------------       /65025,/65280
```

3.7 Meg 16-Bit IIe

Why pay more for a lesser card just because it's advertised a lot? You can buy Checkmate Technology's **State-Of-The-Art MULTIRAM IIe**™ that works great in 8-Bit (100%), has an optional real 16-Bit 65C816 slot saver Co-Processor card, **PORT FOR OPTIONAL BATTERY BACKED-UP MEMORY, FREE RGB,** & in coming months semi-permanent **STATIC RAM options** that can load & save entire programs, like AppleWorks, up to 10 years!

- **MULTIRAM IIe IS A DIRECT SUBSTITUTE FOR RAMWORKS II™ or Apple Ext 80 column cards & fits ALL (even Euro) Apples. It RUNS ALL (100%) 3rd PARTY SOFTWARE that the others do.**
- **TWO MAIN BOARDS TO PICK FROM** both with 16-Bit port, sharp 80 columns & Double Hi-Res Graphics. One with **FREE RGB & STATIC RAM PORT**-expandable to 1024k standard RAM + 2048k BATTERY BACKED-UP STATIC RAM, and one without RGB or port-expandable to 768k. Both cards can piggyback for a **TOTAL POSSIBLE MEMORY OF 3.75 MEGABYTES IN ONE SLOT.** AND ***UNLIKE RAMWORKS II,*** there is never a chance to touch cards in slot one (such as Super Serial Card).
- **OPTIONAL 16-Bit 65C816 CO-PROCESSOR CARD** that plugs into MULTIRAM using no slots! TRULY STATE-OF-THE-ART, BUT IT CAN'T RUN ON RAMWORKS™!
- **FREE APPLEWORKS TIME/DAY/DATE ON-SCREEN SOFTWARE. FREE AUTO-COPY SOFTWARE (disk to RAM). FREE ULTRA-FAST RAM DISK SOFTWARE** that can be run alone or WITH APPLEWORKS. FREE RAM TEST & optional CP/M & Pascal Ram disk ($20. ea)!
- **FREE APPLEWORKS EXPANDER SOFTWARE** that modifies AppleWorks once-loads ALL (even printer routines) or PARTS of AppleWorks into memory, runs 30 x faster, increases the Desktop over 1125k, auto-segments large files onto multiple disks, stores over 5325 records! An increased Clipboard & Word Processor update is due soon.
- **15 DAY MONEY BACK SATISFACTION GUARANTEE,** FREE SOFTWARE UPDATES, FREE 64k MEMORY WITH EACH 256k/512k CARD, & QUANTITY DISCOUNTS FROM COIT VALLEY COMPUTERS. EXCLUSIVE 5 YEAR WARRANTY THAT, UNLIKE RAMWORKS, INSURES COVERAGE NO MATTER WHERE YOU BOUGHT IT! **CALL FOR CURRENT PRICES, QUANTITY DISCOUNTS, NEW FEATURES OR DETAILS ON CHECKMATE'S DIRECT $50 REBATE FOR OLDER CARDS!**

		LOWER OUR ~~LOW~~ PRICE	No RGB
64k	MULTIRAM IIe/FREE RGB	**185.**	147.
128k	MULTIRAM IIe/FREE RGB	**203.**	168.
320k	MULTIRAM IIe/FREE RGB	**222.**	189.
576k	**MULTIRAM** IIe/FREE RGB	**272.**	237.
768k	**MULTIRAM** IIe/FREE RGB	**322.**	287.
1024k	**MULTIRAM** IIe/FREE RGB	**372.**	N/A
1280k	MULTIRAM IIe/FREE RGB	**520.**	N/A
1536k	MULTIRAM IIe/FREE RGB	**548.**	N/A
1792k	MULTIRAM IIe/FREE RGB	**578.**	N/A

256k Memory Expander Chips (8)	**55.**
Apple IIe Enhancement Kit	**62.**
Clockworks (Thunder/Time H.O.™ compatable)	**89.**
FD 100 Slimline Drive IIe, II+	**107.**
65C816 Co-Processor Card	**162.**
Pinpoint Software	**49.**
Disk Mailers (10)	**7.**
Accelerator IIe-speedup card	**249.**

CALL FOR OTHER APPLE PERIPHERAL PRICES.

640k 16-Bit IIc!

Checkmate Technology's **State-Of-The-Art IIc cards easily expand your IIc up to 640k,** are 100% compatable with all IIc software/hardware, & come with the **SAME FREE SOFTWARE as MULTIRAM IIe. MULTIRAM C** is non-upgradable & **MULTIRAM CX** can be upgraded with a **16-Bit 65C816 kit! CALL FOR CURRENT PRICING & QUANTITY DISCOUNTS!**

- **NO JUMPER WIRES, CLIPS, OR DRIVE REMOVAL REQUIRED FOR INSTALLATION. ALL CHIPS ARE SOCKETED AND REMOVABLE**-unlike the competition.
- **USES ABOUT 50% LESS POWER** than the competition-causing less power supply strain or battery pack drain!
- **15 DAY MONEY BACK SATISFACTION GUARANTEE** from Coit Valley Computers. 5 YEAR WARRANTY THAT, UNLIKE THE COMPETITION, INSURES COVERAGE NO MATTER WHERE YOU BOUGHT IT!
- **LOWER PRICES**-We sell IIc cards for much less than the competitor and our software updates are free, while they usually charge $10 or more!

	LOWER OUR ~~LOW~~ PRICE
256k MULTIRAM C	224.
512k MULTIRAM C	264.
256k MULTIRAM CX	278.
512k MULTIRAM CX	318.
16-Bit 65C816 CX Kit ($10 less with card)	135.
IIc System Clock (Same as A.E.)	66.
IIc Battery Pack & U.P.S.	174.
C-VUE Flat Panel Display ($10 less with card)	439.

Terms: Add $4-Ground or $6-Air shipping & phone # to each U.S. order (foreign orders extra). Add 3% for P.O.'s (3% 7 net 30) & MasterCard/Visa **(include #/expir).** For fast delivery send Cashier's/Certified check, Money Order. C.O.D. (add $5) & personal checks accepted (allow 14 days). Tex res add 6⅛% tax. CALL FOR CURRENT PRICES & QUANTITY DISCOUNTS!

MULTIRAM, Ramworks/Ramworks II/Timemaster II H.O., Pico respective trademarks of Checkmate Technology, Applied Engineering, WGE.

COIT VALLEY COMPUTERS (214) 234-5047
14055 Waterfall Way
Dallas, Texas 75240

New Hardware for Programming PALs......Bob Sander-Cederlof

PALs (programmable array logic chips) are to logic circuitry as ROMs are to memory. Most of the new cards coming out these days contain one or more PALs. Engineers write logic equations, feed them into a PAL Assembler, and run the output to a PAL burner. The programmed PAL is then ready to use in a circuit. Until now, you had to buy a PAL development system, either stand-alone or perhaps interfaced to an IBM-alike.

But now, Dynatek Electronics has introduced a new board than slips nicely into an Apple slot for programming 20- and 24-pin PALs. The PALP-701A, for $245, programs 20-pin PALs. The PALP-702A handles both 20- and 24-pin chips, and can also blow the security fuse when you are ready for it. Both of them come with the PAL Assembler software.

Dynatek's PAL Assembler is compatible with Monolithic Memories PALASM. It creates a fuse plot from a PAL source file of Boolean equations. The fuse plot is then used by the PAL Programmer card via on-board firmware to program the PAL. The firmware on the Programmer card can also read un-protected PALs, and verify them. There is also a screen editor for creating, examining, and modifying a fuse plot.

If you design and build circuits, you ought to investigate this card. Call Jerry Wang at (312) 255-3469, or write to Dynatek Electronics, Inc., P. O. Box 1567, Arlington Heights, IL 60006. Tell him we sent you!

Review of Applied Engineering Transwarp.....Bob Sander-Cederlof

We reviewed the M-c-T SpeedDemon accelerator card in AAL of July 1985. At the time the price was $295 from the manufacturer or $199 through Call APPLE. We recently received a promotion sent to software publishers offering wholesale prices if we would advertise the SpeedDemon in conjunction with our software. The suggested price is now $249. (We notice that at least one game publisher took them up on the offer.)

Now Applied Engineering has released their new accelerator card, the Transwarp. Their price is $279 with a 65C02 installed, and an optional upgrade to a fast 65802 for an additional $89. The higher price is probably well justified by the features.

Transwarp includes 256K of high-speed RAM on the card. This compares to 64K on the Titan Accelerator, and a 4K cache on the SpeedDemon. Transwarp will run with the SWYFT card installed, while the others apparently will not.

Transwarp's 256K RAM is effectively divided into four 64K banks. When you power-up your Apple with Transwarp installed, all of the ROM from $D000 through $FFFF is copied into one of the high-speed RAM banks. The rest of this bank is not used. A second bank is used in place of the motherboard RAM. The third and fourth banks are used in place of the first and second banks of AUXMEM, if you have a RAM card such as RAMWORKS installed in the AUX slot. If you have a large RAMWORKS in the auxiliary slot of a //e, any additional banks beyond two will still be usable but at "only" 1 MHz.

When you write data to one of the screen areas (any address $400-$BFF or $2000-$5FFF), the data is "written through" to the motherboard RAM. (The video hardware in the Apple requires that the screen data be in motherboard RAM.) When you read from any of these addresses, the data will be read from the fast Transwarp RAM.

Transwarp keeps track of the state of all the AUXMEM soft switches, as well as the RAMWORKS bank register. All reads from any memory that is supported in the Transwarp RAM will be done at full speed. Reads from and writes to any address in the range $C000-$CFFF will slow down to 1 MHz for one cycle.

There are 16 dip switches on the card, allowing you to configure for most environments. Seven switches indicate which slots must execute code at 1 MHz. Slots designated by switches will slow down the processor for about 1/2 second after any access to either the slot ROM or the slot registers. An Apple disk Controller must run at the slow speed, while most other slots can run faster. Some I/O cards, especially serial cards, must run at slow speed due to internal software-controlled timing. The Transwarp's switches are much more flexible than the SpeedDemon's system of always slowing down for slot 6 and using jumpers to allow a slowdown for slots 4 and 5.

Another seven switches let you indicate which slots (1-7) have

RAM cards installed. The two remaining switches let you select the initial speed of the Transwarp card. You can select a default speed of 3.58 MHz, 1.7 MHz, or 1 MHz. This is the speed the card runs at when you power up. You might like the 1.7 MHz speed for making your game software just a LITTLE faster.

Once the Transwarp has taken over, you can switch back and forth between the default speed and 1 MHz by storing either 0 (default speed) or 1 (1 MHz) into $C074. In BASIC this would be POKE to -16268 or 49268 of either 0 or 1.

If you POKE a value of 3 to $C074, Transwarp will be shut down completely; the motherboard processor will take over when you hit CTRL-RESET. In order to turn Transwarp back on, you have to turn the computer off and back on again with the power switch. You also have the option of disabling Transwarp during the power-on cycle, by typing the ESCAPE key within a couple of seconds after turning on the computer.

Transwarp has a 4K EPROM on-board with startup and self-test firmware. Naturally, I disassembled the code to see how it all works. The self-test is initiated by typing a "0" or "9" during the first two seconds. The test checks for the type of processor installed (65C02 or 65802), measures the speed, tests bank switching, and tests RAM. If you are in a //e, you can hold down the Open-Apple key to keep it looping through the speed test.

Transwarp measures its own speed by counting how many cycles it takes for the Vertical Blanking Signal to pass by. This signal is not available on the II or II Plus, so no speed information is tested on the older machines.

We tested Transwarp doing various jobs such as assembling, word processing, and spreadsheet-ing. Everything worked, no glitches, and a lot faster. The speedup factor depends on the amount of disk I/O, screen I/O, and so on. Nothing runs with a full 3.5 or 3.6 speed increase, not even a short timing loop. The very highest factor I could coax out of my board was about 3.3, on a timing loop running at $C00. This loop included a large number of STA instructions, on purpose. When I moved the program to $800, so that the STA instructions were storing into the range slowed down to 1MHz (between $400 and $BFF), the loop only ran 2.0 times faster under Transwarp than under a normal 1 MHz processor.

Why do the advertisements for accelerators claim a 3.6 or larger speedup factor? I think they are rounding up the clock speed of 3.579... to 3.6, and likewise rounding down the Apple's clock speed from 1.023 to 1. That is not the way the IRS likes you to do math.... The actual ratio of the two clock speeds is exactly 3.5, but the mist does not entirely clear yet.

Remember that the Apple stretches one cycle out of every 65 by an amount equal to one cycle of the 7MHz signal. See chapter 3 of Jim Sather's "Understanding the Apple //e" for details.

With Z-80 Plus,™ run CP/M® – the largest body of software in existence.

Now, get two computers in one, and all the advantages of both.

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering, and introduce your Apple IIe® or II + ® to the thousands of CP/M programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system ever for running CP/M programs.

The new 4.0 boasts advanced features like built-in disk emulation for popular memory expansion boards, boosting both system speed and storage capacity. And menu-driven utilities that let you get to work faster. The Z-80 Plus also lets you run older CP/M programs — all the way down to Version 1.6 (2.2 is the most popular).

The Z-80 Plus is the only card on the market capable of accessing more than 64K in an Apple IIe. If you have an extended 80-column card, all 128K is usable, and if you have RamWorks, up to 1088K is available.

Each Z-80 Plus comes with our CP/M Ram Drive software, enabling IIe owners to use an extended 80-column card or a RamWorks card as a high-speed Ram disk which runs CP/M software up to *twenty times faster.* So packages like WordStar and dBASE II run at blinding speed.

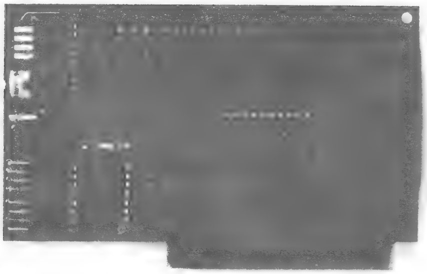
Simply plug the Z-80 Plus into any slot in your Apple. You'll get the benefits of two computers in one — all at an unbelievably low price (only $139!).

• Fully compatible with ALL CP/M software • Fully compatible with most hard disks, including Corvus and the Sider • Fully compatible with Microsoft disks (no pre-boot required) • Specifically designed for high speed operation in the Apple IIe (runs just as fast in the Apple II + and Franklin) • Runs WordStar, dBASE II, Turbo Pascal, Fortran-80, Peachtree and ALL other CP/M software with *no pre-boot* • Semi-custom I.C. and low parts count allows Z-80 Plus to fly through CP/M programs with extremely low power consumption (we use the Z-80B) • Does EVERYTHING other Z-80 boards do, *plus* Z-80 interrupts • Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Timemaster H.O.,™ the only clock that displays time and date on AppleWorks'™ screens and files.



Now, get all the features of all the competition combined!

It's the smart way to put the time and date on your Apple II + ® or IIe® Because only the Timemaster H.O. packs *ALL* the features of all the competition *combined,* including leap year, year (not just in PRO-DOS), month, date, day of week, hours, minutes, seconds and milliseconds. It's totally PRO-DOS, DOS 3.3, PASCAL and CP/M compatible. And of course, it works better than any other clock with AppleWorks.

If you're using or writing software for other clock cards, you're still covered. Because the H.O. will *automatically* emulate them. And the Timemaster H.O. adds 14 new commands to BASIC. The H.O. even comes complete with two disks full of sample programs, including a computerized appointment book, a DOS dating program, interrupt programs, and over 30 programs that others charge extra for — or don't even offer.

As a low cost option, you can add true BSR remote control to the H.O., giving you remote control of up to 16 lights and appliances in your home or office.

• Fully PRO-DOS and DOS 3.3, CP/M and PASCAL compatible • Time in hours, minutes, seconds and milliseconds (the ONLY PRO-DOS compatible card with millisecond capability); date with year, month, day of week and leap year • 24-Hour military format or 12-hour AM/PM format • Eight software controlled interrupts so you can run two programs at the same time (many examples included) • Allows AppleWorks to time and date stamp all data automatically • The only clock card that displays time and date on the AppleWorks screen • Five year warranty

Clock price	$129.00
BSR option (may be added later) .	$ 49.00

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

The original Apple II ROM has executable code at $FB09, and in hex it is this: B0 A2 20 4A FF 38 B0 9E. All other Apple monitor ROMs have an ASCII string at $FB09. The string is either "APPLE][" or "Apple][". Notice that the "AND #$DF" in the checksummer strips out the upper/lower case bit, making both ASCII strings the same.

I wrote a test program to print out all the intermediate values during the "Checksummer's" operation. Here are the results, for both kinds of ROMs.

Original ROM					Later ROMs				
LDA	AND	ADC	STA	ROL	LDA	AND	ADC	STA	ROL
B0	90	00	90	20	C1	C1	00	C1	82
A2	82	20	A2	44	D0/F0	D0	82	52	A5
20	00	44	44	88	D0/F0	D0	A5	75	EB
4A	4A	88	D2	A4	CC/EC	CC	EB	B7	6F
FF	DF	A4	83	07	C5/E5	C5	6F	34	69
38	18	07	1F	3E	A0	80	69	E9	D2
B0	90	3E	C3	9C	DD	DD	D2	AF	5F
9E	9E	9C	3A	75	DB	DB	5F	3A	75

I don't understand why this code gives the same result, but I see it does. Now, dear readers, tell me how anyone ever figured out what sequence of operations would produce the same result using these two different sets of eight bytes, and yet produce a different result for clones! If you understand it, please explain it to me!

By the way, here is a listing of my test program:

```
                  1000 *SAVE TEST.CKSUMMER
                  1010 *--------------------------------
                  1020 *   SIMULATE PRODOS $FB09-FB10 CHECK-SUMMER
                  1030 *       (AT $267C IN PRODOS 1.1.1)
                  1040 *--------------------------------
                  1050 T
0800- A9 68       1060        LDA #S1
0802- 85 0A       1070        STA $0A
0804- A9 08       1080        LDA /S1
0806- 85 0B       1090        STA $0B
0808- 20 13 08    1100        JSR CS
080B- A9 70       1110        LDA #S2
080D- 85 0A       1120        STA $0A
080F- A9 08       1130        LDA /S2
0811- 85 0B       1140        STA $0B
                  1150 CS
0813- 20 8F 08    1160        JSR PT
0816- 18          1170        CLC
0817- A0 00       1180        LDY #0
0819- 8C 67 08    1190        STY X
081C- B1 0A       1200 .1     LDA ($0A),Y
081E- 20 57 08    1210        JSR B
0821- 29 DF       1220        AND #$DF
0823- 20 57 08    1230        JSR B
0826- AD 67 08    1240        LDA X
0829- 20 57 08    1250        JSR B
082C- B1 0A       1260        LDA ($0A),Y
082E- 29 DF       1270        AND #$DF
0830- 6D 67 08    1280        ADC X
0833- 8D 67 08    1290        STA X
0836- 20 57 08    1300        JSR B
0839- 2E 67 08    1310        ROL X
083C- AD 67 08    1320        LDA X
083F- 20 57 08    1330        JSR B
0842- 20 8E FD    1340        JSR $FD8E
```

If you remember Tom's DOSTalk column from the much-missed pages of Softalk Magazine, or are familiar with his current Open-Apple newsletter, you know that what he writes is easy to read, fun to read, and WORTH READING.

Seven fascinating chapters lead you to an understanding of how financial transactions really work. He starts with simple percentage calculations, at a level your Junior High children can follow. If you think that is starting too simply, try explaining percentages to YOUR children! But he keeps going....

Have you thought about buying a house recently? Tom shows you how to figure the true cost of an adjustable-rate mortgage, how to compare different financing schemes, and how to protect your money. You'll learn about the tricks money lenders sometimes use to take advantage of unwary investors and borrowers. And all is tied to spreadsheet models you can put into your Apple. I wish I had only known how to do these things when I bought a pickup truck last summer. Or leased a copying machine three years ago. And when we bought some land in the country....

The book is 160 pages slim (172 counting everything), only $9.95 at your favorite book store. And worth a trip! Or call Gerald Rafferty at Info Books, (213) 470-6786. Or write to them at P. O. Box 1018, Santa Monica, CA 90406.

With these tools you can really program!

Presenting a selection of the best programming tools for the entire Apple II family!

S-C Macro Assembler – Combined editor/assembler includes 29 commands and 20 directives; with macros, conditional assembly, global replace, edit, and more. Well-known for ease-of-use and powerful features. Thousands of users in over 30 countries and in every type of industry attest to its speed, dependability, and user-friendliness. Blends power, simplicity, and performance to provide the optimum capabilities to both beginning and professional programmers. With 100-page manual and reference card. $80. (If you own an older version, call for an upgrade price.) Version 2.0 - $100

Cross Assembler Modules – Owners of the S-C Macro Assembler may add the ability to develop programs for other systems. We have modules for most of the popular chips, at very reasonable prices:

6800/01/02	$32.50
6805	$32.50
6809	$32.50
68000	$50.00
Z-80	$32.50
PDP-11	$50.00
8048	$32.50
8051	$32.50
8080/85	$32.50
1802/04/05	$32.50

All of the cross assemblers retain the full power of the S-C Macro Assembler. You can develop programs for burning into EPROMs, transfer through a data-link, or direct execution by some of the plug-in processor cards now on the market.

Apple Assembly Line – Monthly newsletter for assembly language programmers, beginner or advanced. Tutorial articles; advanced techniques; handy utility programs; commented listings of code in DOS and Apple ROMs; reviews of relevant new books, hardware, software; and more! $18 per year (add $3 for first class postage in USA, Canada, Mexico; add $13 postage for other countries).

ES-CAPE (Extended S-C Applesoft Program Editor) – Full-function, interactive program editor for Applesoft. For painless programming and accurate entry of programs from magazines: fast, easy changes; split-screen operation; global search and replace; automatic line numbers; keyboard macros; and more! ~~$60~~$50 w/source.

Flash! – The Integer BASIC Compiler from Laumer Research. Transforms your programs into machine language so they run many times faster. Adds many new functions and features to the language, supporting hi-res graphics and more! Optional assembly source code creation for use with S-C Assembler. Requires Integer BASIC in RAM or ROM to edit source programs (not required for compiled programs). Source code of run-time package also available. Compiler, $79; run-time source, $39.

Double Precision Floating Point for Applesoft – For the scientist or engineer who is not satisfied with Applesoft's nine-digit precision. With this 2048-byte machine language extension to Applesoft, you can get 21-digit precision whenever you need it! Supports +, −, *, /, INPUT, and PRINT. Subroutines coded in Applesoft for 21-digit precision math functions (sin, cos, sqrt, log, exp, etc.) are also included. And it's fast! Only $50.

DISASM – Symbolic two-pass disassembler handles data tables, displaced object code, and includes a pre-defined symbol table of your choice. Produces ready-to-assemble source code with labels from machine code in ROM or RAM, formatted for S-C, Lisa, or ToolKit assemblers. $30.

S-C XREF – Designed to work with the S-C Macro Assember, produces a cross reference table of all labels used in your assembly source code. Only $20, or $50 with complete source code on disk.

All of our products are designed to be used. You can make back-up copies with any standard Apple disk copy program, and copies of the individual files can be moved to your working disks. We trust you.

S-C Software Corporation has frequently been commended for outstanding support: competent telephone help, continuing enhancements, and excellent upgrade policies. Call now and talk to Bob Sander-Cederlof or Bill Morgan. We want to help you program!

S-C Software Corporation
2331 Gus Thomasson, Suite 125
Dallas, Texas 75228
(214) 324-2050

We take Mastercard, Visa, and American Express

Apple is a trademark of Apple Computer

Which Processor Am I In?......................Jim Poponoe

One of the first programs I wrote after receiving my 65802 chip was one which tells me which microprocessor is in my Apple. Since the 65C02 has instructions not in the 6502, and since the 65802 has all of those and still more, it is possible to tell which is which.

The instructions in the 65802 (or 65816) which are not in the 65C02 are all "no-operation" opcodes in the 65C02. The same is not true for the un-implemented codes in the 6502! Bob S-C detailed what all the un-implemented 6502 opcodes do in the March 1981 issue of AAL. Some of them do really exotic things, but some are in fact NOPs. $80 is a two-byte NOP in the 6502, but a Branch Always (BRA) in the 65C02 and 658xx. Therefore, the BRA opcode can be used to distinguish between the 6502 and higher versions.

The XBA instruction ($EB) is a one-byte no-operation in the 65C02. In the 658xx it exchanges the low and high bytes of the 16-bit A-register. Therefore it can be used to distinguish between the 65C02 and the 658xx processors.

The following program will print out either "6502", "65C02", or "65802" depending on which it finds. A few more tests could distinguish the Rockwell 65C02, which has four opcodes beyond those in 65C02s made by other manufacturers. And a few more might distinguish between a 65802 in my motherboard and a 65816 running in a co-processor card. I'll leave those for interested readers to try.

```
                         1000 *SAVE S.WHICH PROCESSOR
                         1010        .OP 65802
                         1020 *--------------------------------
FDDA-                    1030 PRBYTE .EQ $FDDA
FDED-                    1040 COUT   .EQ $FDED
                         1050 *--------------------------------
                         1060 WHICH.PROCESSOR
000800- A9 65            1070        LDA #$65
000802- 20 DA FD         1080        JSR PRBYTE
000805- 80 03            1090        BRA .1
000807- 4C 13 08         1100        JMP .2
00080A- A9 B8            1110 .1     LDA #"8"
00080C- EB               1120        XBA
00080D- A9 C3            1130        LDA #"C"
00080F- EB               1140        XBA
000810- 20 ED FD         1150        JSR COUT
000813- A9 02            1160 .2     LDA #$02
000815- 4C DA FD         1170        JMP PRBYTE
                         1180 *--------------------------------
```

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $18 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $14 postage for other countries. Back issues are available for $1.80 each (other countries add $1 per back issue for postage).